Using CT-Analyst® as an integrated tool for CBR analysis

Adam Moses, Keith Obenschain, Jay Boris

Laboratory for Computational Physics and Fluid Dynamics
U.S. Naval Research Laboratory, Code 6400
Washington DC, USA 20375-5344

ABSTRACT

Modern information systems that are designed to plan for and respond to Chemical, Biological, and Radiological (CBR) attacks are now attempting to include airborne contaminant plume models in their application package. These plume models are a necessary component for the emergency personnel responding to an actual event and to those charged with developing an effective response plan in advance. The capabilities to create a variety of CBR-event scenarios quickly, to determine possible contaminant agent release locations from reports and sensor data, and to predict the path of a plume before it gets there, are functions that many involved in the military and homeland security will find beneficial. For this reason CT-Analyst's ability to generate accurate, time-dependant plumes that can be rendered much faster than real-time and can be adjusted and modified on the screen, is an incredible asset to developers of these civil defense systems. The value of the fast, accurate CT-Analyst computer models for complex urban terrain is greatly increased when the capabilities can be imported into platforms users and developers are already taking advantage of. CT-Analyst's strengths can now be accessed through other applications, GIS tools, and development environments. This process will be described to show how this is possible, to which systems it can be applied, and what benefits can result.

Keywords: CT-Analyst, Sensor Integration, CBR, Plume Modeling, Fluid Dynamics, Emergency Response

1. INTRODUCTION

Following the events of September 11th the need for an event planning and response tool, one that can deal quickly and accurately with atmospheric Chemical, Biological, or Radiological (CBR) plumes in urban areas, was clearly visible. The Laboratory for Computational Physics and Fluid Dynamics at the U.S. Naval Research Laboratory had developed and were demonstrating[1] the Contaminant Transport Analyst (CT-Analyst), a product perfect for these tasks. By using the FAST3D-CT code as the detailed basis for analyzing urban contaminant transport, one could simulate the release of any airborne CBR agents. The capability to perform these detailed runs allowed the creation of fast transport and dispersion models based on Nomograf™ databases. These models[2], used exclusively in CT-Analyst, create visualizations in milliseconds for a variety of important displays including contaminant release footprints, escape paths, upwind danger zones, and sensor consequence areas. CT-Analyst also incorporates "backtracking" to establish the location of covert sources and to optimize the placement of sensors, among a number of important functions.

While this application has been developed into an available product for operational deployment today, using it as an integrated tool embedded in systems that perform full spectrum CBR crisis and assessment functions is a key step for the future. This integration process is the focus of this paper. The discussions are from both a design and a development perspective and showcase a few areas where CT-Analyst has already been well applied.

1.1 CT-Analyst Background

The genesis of CT-Analyst came out of the Department of Defense's High Performance Computer Modernization Program (HPCMP). HPCMP wanted to make the original FAST3D Computational Fluid Dynamics (CFD) code, designed to model airflow over Navy ships at sea, more efficient on large parallel computers for use in a wide range projects. Part of this effort enabled more accurate simulations of vortices interfering with aircraft operations, and later the dispersal of hot containment exhaust fumes. A natural extension of these simulations was applying these models towards CBR contaminant transport, specifically for buildings and urban land formations. This modified form of FAST3D, FAST3D-CT became the basis for the urban environment modeling of CBR plumes. The 3D FAST3D-CT

code accurately models complex building geometry, vortex shedding created by buildings, and the turbulence and re-circulation responsible for the dispersion found in urban environments. It is highly accurate for calculating the aerosol and droplet physical processes, and takes into account solar heating affected by season, time-of-day, and anisotropic heat absorption resulting from shadows produced by buildings and trees. All FAST3D-CT requires is a geometric database of all building, tree, and terrain heights, as well as the bodies of water that are present.

After the FAST3D-CT code has completed the CFD modeling for 18 wind directions in a given urban region that is a few kilometers on a side, the resultant contaminant flow data is processed into CT-Analyst Nomografs. Nomografs drive the plume models that appear inside CT-Analyst and are visually realized by its Graphical User Interface (GUI). More importantly, Nomografs serve as the backbone for the CT-Analyst interface that acts as the engine for the entire software system. This interface is central in exposing the power of CT-Analyst so it can be acted upon by a number of current and potential, Application Programmer Interfaces (APIs). These APIs allow the achievement of a variety of useful tasks that can otherwise be too difficult for a single operator to perform by hand. These tasks, and their integration into other systems, enable CT-Analyst to become an even more powerful tool for CBR analysis.

1.2 Previous CBR Analysis and System Integration Example

While there are not many examples of previous work that mirror what is being attempted here, there are a few. To cite at least one as a helpful background, there is the M39A1 Fox Nuclear, Biological, and Chemical Reconnaissance System (NBCRS), a military vehicle for entering into suspected contaminated combat areas and report back sensor readings. This vehicle's complicated software suite was required as an addition to the Close Combat Tactical Trainer[3] (CCTT), an Army collective training simulation tool. The addition of this new system into CCTT was difficult as it had no previous ability to handle data related to Weapons of Mass Destruction (WMD). For example, wind speed, despite being a part of the system, had never been used before. Much of the underlying system had to be re-worked in order to add the capabilities to the simulator, which posed a challenge to the integrators. The lessons learned here were to keep as much modularity as possible in any design. A focus should be that exposing as many methods as possible is a good thing, but requiring as few as possible to be implemented for use is best.

2. DESIGN

The idea of making CT-Analyst a tool more open for other programs to exploit as a sort of engine required a clear set of goals to be achieved. The outline of the three primary goals that were to be reached were: 1) separate the external interface from the internal workings of CT-Analyst, 2) ensure a robust system able to replicate as much of CT-Analyst as possible, and 3) make sure that interface contained sufficient flexibility to work with as many other tools and systems as could be imagined.

2.1 Goals

The goal of separating the internal CT-Analyst functionality from all external operations is based on several considerations. Foremost, avoiding modification of the internal operation while exposing only what is actually needed by potential users makes the product far more reliable to client developers. Since the external interface and internal operations are bound in separate spheres of operations, the task of responding to future needs and executing fixes for current users is made much simpler for everyone involved. If client developers modified code to specific internal functions, it can be far more difficult to make beneficial updates in the future and still ensure everything worked as it had before.

Of comparable importance is the goal of allowing as much external access to CT-Analyst functions as possible. Client developers presumably will have experience with the standalone CT-Analyst. If they use a capability found there and it is not available through the external interface, the overall system can be less attractive for them to use. The design absolutely must allow a headless version of the external interface, one that lacks the GUI. The external application programming interface is, by its very nature, required to mimic everything that is possible through regular keyboard and mouse interaction. It was important that the external interface maintain as many methods and properties as possible, guaranteeing client developers a wide a range of variant programmatic combinations.

Finally, keeping the external interface open-ended for the client systems and tools it will encounter is central to its broad adoption. The vast landscape of programming communication standards available today, such as Common Object

Model (COM), Common Object Request Broker Architecture (CORBA), Simple Object Access Protocol (SOAP), and others, requires the external interface to "play" with as many systems and tools as possible. This does not include the collection of government and military standards and systems like High Level Architecture / Real Time Infrastructure (HLA/RTI), Global Command and Control System (GCCS), and Virtual Mission Operations Center (VMOC), some of which may be considered in the future. Indeed the current design makes these extensions straightforward.

2.2 Component Breakdown

As a part of the design to fulfill the goals just mentioned and to keep in step with concurrent CT-Analyst development, it is critical to adopt a component structure that maintained a specific rule set defining which process could execute where. This design required us to clearly allocate certain operations to specific workspaces and to establish clear lines of demarcation between them.

The way in which the core components are encapsulated in their own area of operation is illustrated in Figure 1. The most critical component shown is the CT-Analyst Interface, also referred to as the Internal Interface for much of this paper. This interface contains all the control mechanisms, data structures, and class definitions that the remainder of the application relies on. The Nomograf Library component is responsible for reading Nomograf files and managing their processing in order for the Internal Interface to take advantage of the data resident in them.

Figure 1 also shows two tightly coupled components, an important feature of the CT-Analyst architecture. These components are the GUI and Sensor Placement components. The GUI is the interface to interactive users and the Sensor Placement component is used as part of a separate genetic algorithm module[4], one capable of optimizing sensor placement for any given region where the Nomografs have been pre-computed. The purpose of the tightly coupled components is to function very closely with the Internal Interface for efficiency. This tight coupling is somewhat restrictive to outside developers and is thus outside the scope of the full API. These two components are distinguished from the other integrations being discussed here that act through the External Interface, the remaining internal component requiring discussion. Mentioned already as the design goal of this application, the purpose of the External Interface in design terms is to act as the conduit through which all client system integration will occur. Being separate from the Internal Interface, the External Interface is modular, enabling other developers a standard way of interacting with all current and future aspects of CT-Analyst available to users through the GUI. Its own close ties with the CT-Analyst Interface also ensure that reproducing the functions of the central engine remains a smooth process.

The remainder of Figure 1 describes the client systems with which integration is possible, how the External Interface will communicate with these systems, and the actual constructed examples of how this has been implemented. In the Integration section of this paper, the COM component, the feature component actually deployed and in use today, is highlighted. There are also other numerous client communication systems, including, but by no means limited to, HLA/RTI, CORBA, and Web Services.

3. DEVELOPMENT

Taking the External Interface from a design perspective to a fully realized tool for integration with a CBR Analysis application, developers first need to choose which system to create the application under. As well, developers have to understand what is needed as a part of the system, specifically what to access from the CT-Analyst Interface. These requirements entailed the creation of the CT-Analyst Application Programmer's Interface (API) that defines and maintains the ground rules for how the CBR application will access and interact with CT-Analyst, and what information is passing in and out of it.

3.1 ctaCOM

Considering systems available to implement the External Interface, the Common Object Model (COM) seemed to be the best fit for the first implementation. COM is a critical part of the Microsoft world, and thus works under a Windows environment - the most widely used operating system. The choice of an operating system and component type that so many are already familiar decidedly made COM the way to go.

An instance or usage of COM, known as a COM object, is essentially a single execution of that code on a given machine. COM objects are discrete components[5], which expose interfaces that allow applications and other components to access their features, each with a unique identity. More versatile than Win32 DLLs because they are completely language-independent, COM objects have built-in inter-process communications capability, and easily fit into an object-oriented program design. The COM object for CT-Analyst is named ctaCOM.

Much of CT-Analyst is written under the QT Framework, a product of Trolltech, so COM continued to be a desirable choice. An entire facet of QT, known as ActiveQT, is explicitly written for importation into a COM paradigm, streamlining the initial development. The advantage of ActiveQT is that it allows developers to write code in C/C++, the same language used in the Internal Interface layer of CT-Analyst. During compilation, ActiveQT automatically generates all libraries and executables needed to run the tool. ActiveQT is also capable of displaying the tool as a widget, allowing developers to maintain all the GUI elements found in CT-Analyst, as well as providing a headless, engine-only version.

Finally COM continued to be a good choice because of how closely linked it is to the Windows operating system. This characteristic makes for quick usability under almost any development environment available today, and in any project developers may design. In many projects it's as simple as adding the ctaCOM object to a toolset, and performing a drag and drop into the work area. Considering also the collection of Windows ActiveX/COM tools available, having ctaCOM work along with them is thus simplified. Each of these points in support of COM made it the obvious choice for the first implementation of the CT-Analyst External Interface.

3.2 The API

After choosing COM, the next logical question was what to put into it. The internal workings of CT-Analyst are numerous, and only a few of the method calls need be invoked for use by outside developers. A goal was to maintain separate workspaces for the Internal and External Interfaces so it was crucial to ensure that this did not affect the robustness of the set of calls that needed to eventually be allowed.

Possibly the most important question to answer was how to send information to the method calls and to identify what values those calls are returning. As COM is a client library, developers can be unable to accurately debug its internals. This requires the ctaCOM object itself to communicate any problems back to the developer. As a result all calls needed to return an integer error code to the user. The set of methods made available must be able to create new sources, sensors, and sites for CT-Analyst to process, and these objects all needed the use of a unique identifier. Therefore the return value will have to be able to contain this identifier as well as any of the error codes. This then requires only that negatives denote errors and positives denote object identifiers.

Another principle question is the way in which to separate method calls. ctaCOM is just another object requiring a fixed number of method calls for outside developers to work with. This fact exists despite ctaCOM actually representing the entirety of a running application, and a developer may want to access the sub-components and create their own copies of them. Due to certain limitations of the COM paradigm this is not possible, requiring the method calls to be written as simple as possible, reducing the overall communication load. In doing this the calls can remain clearly defined to the client developers, and in combination will still allow for CT-Analyst replication.

The CT-Analyst External Interface should also classify its method calls into groups based on the operations they perform. The most primary method group should encompass the functions that entail all of the CT-Analyst command operations. These include such operations that are necessary for retrieving a new set of Nomografs, switching between Nomografs for different wind directions, or loading or saving a scenario from the current ctaCOM instance. The second operation grouping relates to the GUI elements of the COM widget. GUI operations for the object as a whole are provided by default, so only those parts of the CT-Analyst interface need be mentioned, in this case setting visibility of property and Nomograf panels.

An important facet of CT-Analyst is its precise control over objects in the environment, and the environment itself. Operations related to the environment include those that activate or deactivate the displays, e.g. Footprint display and Escape Paths, as well as controlling the wind and simulation settings. Object operations include those that concern

adding or deleting objects, manipulating objects by moving them or changing their attributes, or reading property values, like particle density from a sensor object.

For easy use by developers, it was important to allow for a quick way to quickly gather array data from the Nomograf display area. These operations require a set of array controls to gather the variable values at all cells located inside the Nomograf region, and return the array to the client system. A limitation in using COM is that it cannot return multi-dimensional arrays, instead only allowing single-dimensional arrays. This requires all data to be transformed into an appropriate linear array and passing associated index values along with it.

Finally, it was desirable to provide a set of event operations attached to ctaCOM so that each setting change in the display area spawns a notification to the client system. Developers use events as a key element to keep programs in sync with one another, as well as allowing a way to monitor results as they come in. These operation groups are sufficient to satisfy all the design goals that were laid out earlier.

3.3 Implementation

Carrying the API from design into an implemented ctaCOM object was only a matter of tying the internal and external components together. This meant making sure every get and set, particularly for objects, was properly attached to its respective function. It also meant checking that any additional operation inserted into the External Interface was functionally possible from what was already available in the internal code. For instance many of the events, operations used for notification, could not be implemented initially since they required considerable overhaul to the way objects were referenced internally. Identifiers were unique but not universal, so to keep the External and Internal Interfaces in sync required changing much of the internal operations.

Several of the steps needed to make the External Interface function also improved CT-Analyst as a whole. One of the important generalizations was how to deal with different coordinate systems. Internally all coordinate space for a Nomograf is cast in a simple Cartesian coordinate system containing the specific region of interest (only a few kilometers on a side) but not referenced to any global coordinate system. Therefore all objects and Nomografs were modified to work in the Universal Transverse Mercator (UTM) Coordinate system, an alteration that simplified things overall. With all the internals working in UTM space, the API was amended to make sure all coordinates expressed there also were working under UTM. The External Interface simplified the addition of a coordinate translation layer, allowing the client system to work under whatever coordinates it desires while leaving the internals of CT-Analyst unchanged.

Some aspects of the implementation proved difficult, requiring modifications and extensions of CT-Analyst to ensure reliability. For example, a new XML-related class was added to simplify and streamline the scenario file format that was developed checkpoint and recall the intermediate state vectors of a particular scenario. To co-exist with this new component, a region information class was also introduced to complement any Nomograf related information particular to a region. Following the initial implementation of the API, extensive testing was performed to guarantee that the translation from ActiveQT into a usable COM object was functioning as expected. This testing will continue as new applications and features are added. The next steps were to conduct trial integrations of CT-Analyst through ctaCOM to demonstrate proof-of-concept for actual and potential outside developers.

4. INTEGRATION

The use of COM objects in Windows and in particular its feature Integrated Development Environment (IDE), Microsoft Visual Studio, made a lot of this work straightforward in the initial ctaCOM implementation. The following are a few examples of the integration work that was done.

4.1 Demo Application with Microsoft Office Integration

The first example of using CT-Analyst as an integrated tool via the ctaCOM object was built under Visual Basic and used the Microsoft Office objects native to it. The short-term goals were several-fold. The first was to construct an example for future developers of how ctaCOM could be used as a coding tutorial. The second was to develop examples of the complex scripting that are possible with ctaCOM, beyond anything that can be done with the GUI. The third goal

was to illustrate Visual Studio's flexibility in incorporating a variety of Windows objects, in this case treating both ctaCOM and the Office objects alike.

The goal for the demonstration application was to showcase all the operations available in ctaCOM. This functionality includes adding the three native CT-Analyst node types, sources, sensors, and sites remotely, then moving them, reading and modifying all the state-vector values associated with them. The functionality also includes controlling the source plume, sensor, and site displays and all the simulation settings such as wind direction and speed. These basic operations were easy to implement, but the need for an additional facility quickly became apparent. With the ability to create complex scripts, there were also a number of calls that needed helper functions. By enabling the client system to make all of its calls through a single set of functions, it became possible to intercept and process error messages produced during a call. This is a critical consideration to note, since other projects may also want to include this in their code.

Scripting is one of the most important capabilities to demonstrate in this project. Since the CT-Analyst GUI is shown through the ctaCOM object as a widget, visualizing the relationship between demonstration application operations and their effect on the CT-Analyst displays is automatic. Several complicated example scripts that perform operations rapidly and apparently at random, were an easy construct. This is also a good way to show how much data CT-Analyst can handle, proving that a seeming overload of commands, such as in a server application, can be handled satisfactorily. In these demonstrations a couple of seconds of executed commands corresponds to hours of computation and waiting with other DoD and DHS models.

The scripting API demonstration shows integrating CT-Analyst with other tools enhances the ability to perform CBR analysis. Microsoft provides tools familiar to most users, and the "Excel" spreadsheet and charting abilities are valued by many. Therefore one ctaCOM demonstration has CT-Analyst rapidly produce a contaminant transport scenario for a complex geometry urban area, record the results, and export the resulting data directly into Office products for immediate presentation and inspection.

The two figures, Figure 2 and Figure 3, illustrate the way in which ctaCOM integration has been applied. In Figure 2 the demo application can be seen replicating in a separate, potentially "remote" window all the basic operations also available through the tightly coupled CT-Analyst GUI. Remotely emulating the Internal Interface, as demonstrated, is one of the primary goals of this project. In Figure 3 the integration, specifically with Office's spreadsheet and charting functions, yields visualized data from CT-Analyst that CBR analysts should find highly useful. Combined with the scripting capabilities, many different sequences of scenarios can be created, yielding a variety of different and interesting results. For example, a path through the domain such as a parade route can be defined and the variation of the contaminant concentration for a given environment and source along that path can be plotted automatically. Simple changes to the scenario, such as dragging the source to another location, immediately yield updated plots in less than a second. Through other CBR plume tools a corresponding demonstration can take hours.

4.2 CoBRA

A goal for ctaCOM development was that CT-Analyst should work with third party CBR systems. The effort with Defense Group Incorporated (DGI) on their Chemical or Biological Response Aid[6] (CoBRA) was a good fit. CoBRA is a toolset designed for use by first responders, law enforcement, Hazardous Materials (HAZMAT) cleanup crews, and even after-action forensic teams. It encompasses a suite of CBR related tools including reference materials on major Chemical and Biological agent threats, checklists and guides for responding onsite to an attack, explosion and blast effect classifications, and perhaps most key, an incident reporting system for sharing data between crews. DGI wished to include CT-Analyst as another component within this suite, finding it a nice fit and extending what their package already offers. The goal was for CT-Analyst to be integrated seamlessly with the rest of the package, making use of as much of the CoBRA specific functionality as possible.

In explicit design terms this meant making sure CT-Analyst menu options are tightly integrated with the GUI system in CoBRA, and that CoBRA is capable of capturing data from CT-Analyst then including this information as part of the incident report system. For reference, an incident report is a way to capture a current snapshot of whatever was going on in a CoBRA tool at that moment. For CT-Analyst this means the scenario currently being displayed including all objects, display options, and timing and wind setting variables.

Connecting CT-Analyst to CoBRA through ctaCOM was complicated by the fact that the CoBRA software itself was undergoing significant changes from version 3 to 4. This was a big step for the CoBRA developers since it involved a major overhaul to the underlying workings of their application, in particular unifying all the various components under a single library to be invoked by the suite of integrated tools. Incident reporting operations are tied to the core system so that tools can be invoked when a report is activated, and data imported into the system at that time. For CT-Analyst, this operational procedure required exporting the entire CT-Analyst scenario file into a CoBRA report, and then activating that report again later, importing the data to recreate the parent CT-Analyst scenario.

Another design goal of DGI, taken from the requests of many of their clients, was to create a list of fixed sites, buildings and other high-value installations in the domain of interest. This list could be extended but the users could not be allowed to move these sites as was possible with all regular CT-Analyst objects. This capability forms the basis for an alert system where a release can be assessed in the case of a real event and all of the fixed sites falling within the instantaneous plume envelope show up in a separate display, complete with reference data. In this way the system, as envisioned, can alert the sites, communicating a timely message to that location's security infrastructure so proper actions can be taken.

Figure 4 shows the integration of ctaCOM into the CoBRA shell. In this figure, the left hand side displays the menu selection bar where the user has chosen CT-Analyst. This selection activates the COM object, and then automatically loads the Nomograf region, as well as any sites found within the region that appear in the alert list. This alert list is detailed in Figure 5, revealing how only the sites within the plume footprint are singled out and listed with pertinent ancillary data about those locations. This is an important CBR analysis feature for users of CoBRA, allowing them to visualize scenarios in simulation mode instantly and to respond to a actual event using the alert system coupled to the incident reporting system to communicate between all the active CoBRA systems.

4.3 ESRI ArcGIS

ArcGIS[7] is a Geographic Information Systems (GIS) application used by developers worldwide. Using it to complement ctaCOM seemed like a natural idea, as its ability to perform complex GIS functions overwhelms nearly all other software like it. Just as ctaCOM and Microsoft Office function as deployable COM objects, ArcGIS also contains COM extensions that allow external replication of much (but not all) of what the total application suite is capable of. Using this object allows the display of an underlying map and then renders vector data over it.

This application required a little translation to achieve the desired result. The ctaCOM object natively exports plume and other displays as array data, but not as a shapefile, the vector format ArcGIS requires. Using other tools, including the GDAL GIS library, with some additional programming it was possible to go from the array data in CT-Analyst to a geo-referenced collection of points describing the boundary area of the plume. Scripting the series of translation steps, and tying this into the ctaCOM event system, allowed automatic updates to the plume each time it was modified.

Figure 6 shows a release footprint plotted from ctaCOM on the left side, and the resultant plume outline as it appears inside the integrated ArcGIS tool on the right. ArcGIS can control the transparency, color, etc. of the display and multiple concentration contour levels can be added with the plume density display, rather than the footprint is selected. This example clearly indicates the wide potential the ctaCOM object has for a variety of different uses. The ability to export plume data to other applications with very few intermediary steps, is perhaps one of the most attractive features for outside developers. For this example alone, the ability to push usable footprint and concentration data, with full urban geometric fidelity, to ArcGIS opens the door to thousands of GIS developers already familiar with ArcGIS.

5. SUMMARY

This paper discussed the design and initial implementations of CT-Analyst integration into other applications and other software tools. This is, however, just the beginning. The range of competing communication protocols with which to export CT-Analyst functionality goes well beyond COM. These additional "standards," as well as additional enhancements to ctaCOM will most certainly enable new ways for developers to take advantage of the near instantaneous plume computation (milliseconds) and Computational Fluid Dynamics-level accuracy made possible by the Nomograf data structure underlying the CT-Analyst engine. Likewise, the number of applications in the CBR field is expansive, and as the already in service CoBRA reveals, including CT-Analyst in a product can be a key enhancement

for existing systems. As part of the Naval Research Laboratory program, the key impediments to universal application of these breakthrough technologies is addressed – the relatively long delay to compute the Nomograf tables for a new region. The solution is a turn-key system where a high-resolution CT-Analyst treatment of an urban region can be produced automatically in a few hours to a couple of days once a GIS-quality urban geometry for the region is available.

What is gained substantively from these first explorations is quite important but this has also clearly been a learning process for everyone involved. All the initial goals discussed at the beginning of this paper were eventually met, but in future endeavors there will be additional goals and new problems encountered as a result. One such crucial research application is the construction of a sensor-information concentrator to automatically evaluate sensor output and report that output to CT-Analyst once a sufficient level of concern is determined. Also planned are new capabilities to analyze the uptake of contaminant by specific buildings ("sites") and the display of optimal response suggestions based on quantitative trade-off of possible sheltering-in-place and evacuation options. The geometric capability of the CT-Analyst engine is being extended to instantaneous line-of-site visibility analyses to support development of a source backtrack capability for standoff sensors to complement the existing, efficient backtrack function for point sensors and reports. In simply constructing the ctaCOM as it now stands, many hurdles had to be dealt with. Also discovered in the process are elements that need additional consideration again in the future. Perhaps what was most valuable to learn was just how powerful the integration of disparate tools can be.

ACKNOWLEDGEMENTS

The authors would like to thank the Naval Research Laboratory for sponsoring this work and providing the computers, software, and research environment that made this work possible. Thanks are offered to Don Ponikvar and the technical staff of Defense Group Incorporated for their interest in and help with the CoBRA interconnection example cited above. Also deserving thanks are Gopal Patnaik, Russell Dahlburg, Ted Young, and Lisa Williams for help and guidance in constructing this paper.

REFERENCES

1. J. Boris et al., *The Threat Of Chemical And Biological Terrorism: Preparing A Response, Computing And Science In Engineering*, pp. 22 - 32, March/April 2002
2. T. Young, Jr. et al., *Emergency Assessment with Sensors and Building: Advances in CT-Analyst Technology, Proceedings of CBIS 2004*, Williamsburg, VA, October 19-20, 2004
3. David M. Jodeit et al., *Integration of Chemical and Radiological Weapons Effects and Sensors into CCTT*, Simulation Interoperability Workshop, March 2002
4. J. Boris et al., *Using CT-Analyst to optimize sensor placement*, SPIE Defense and Security Symposium, Orlando, FL, April 12-16, 2004
5. Jupitermedia Corporation, *Common Object Model*, http://www.webopedia.com/TERM/C/Component_Object_Model.html, Darien, CT
6. Trolltech Incorporated, *ActiveQT*, http://doc.trolltech.com/4.1/activeqt.html, Palo Alto, CA
7. Defense Group Incorporated, *Chemical Biological Response Aide*, http://www.defensegroupinc.com/cobra/, Alexandria, VA
8. Environmental Systems Research Institute, *ArcGIS*, http://www.esri.com/software/arcgis/, Redlands, CA

FIGURES

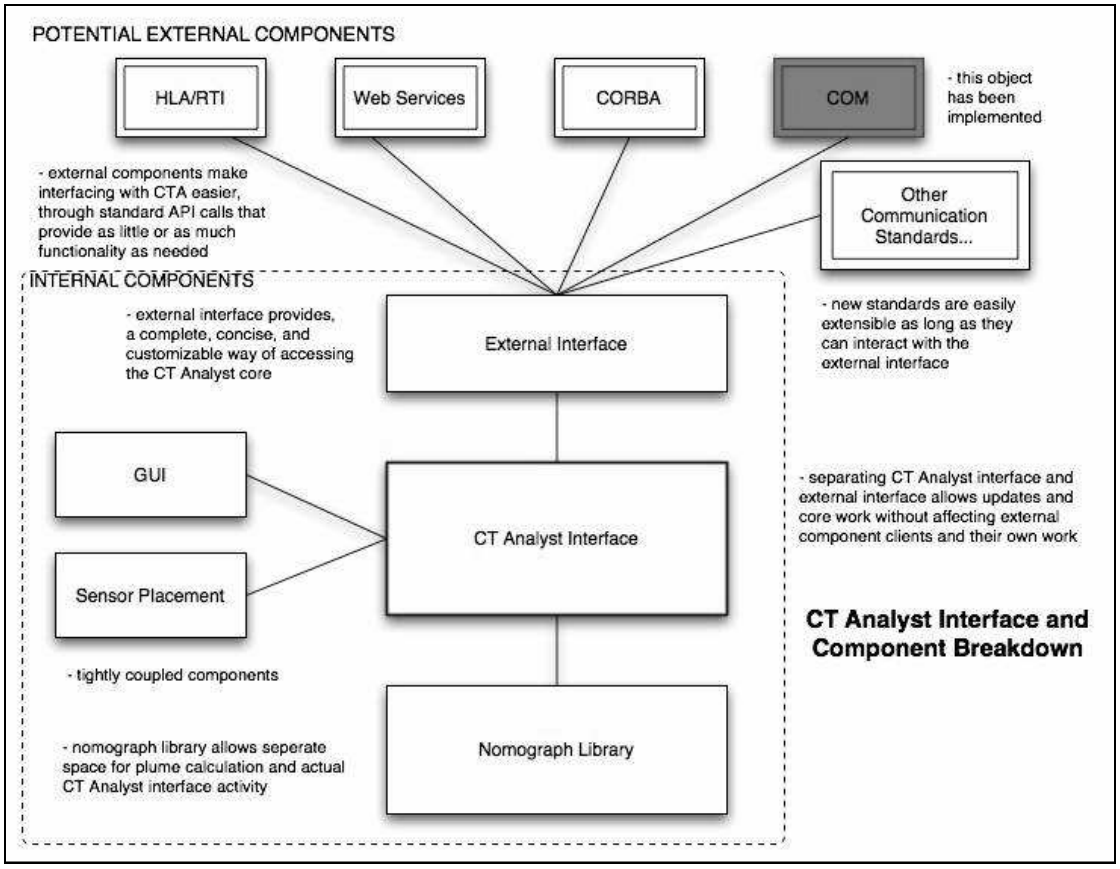


Figure 1: CT-Analyst Interface and Component Breakdown

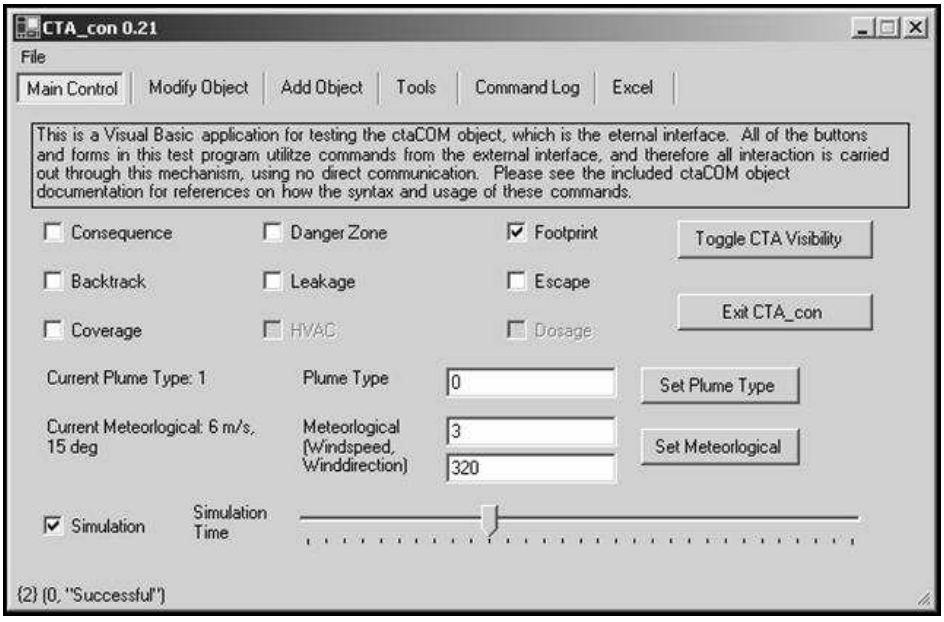


Figure 2: Demo Application

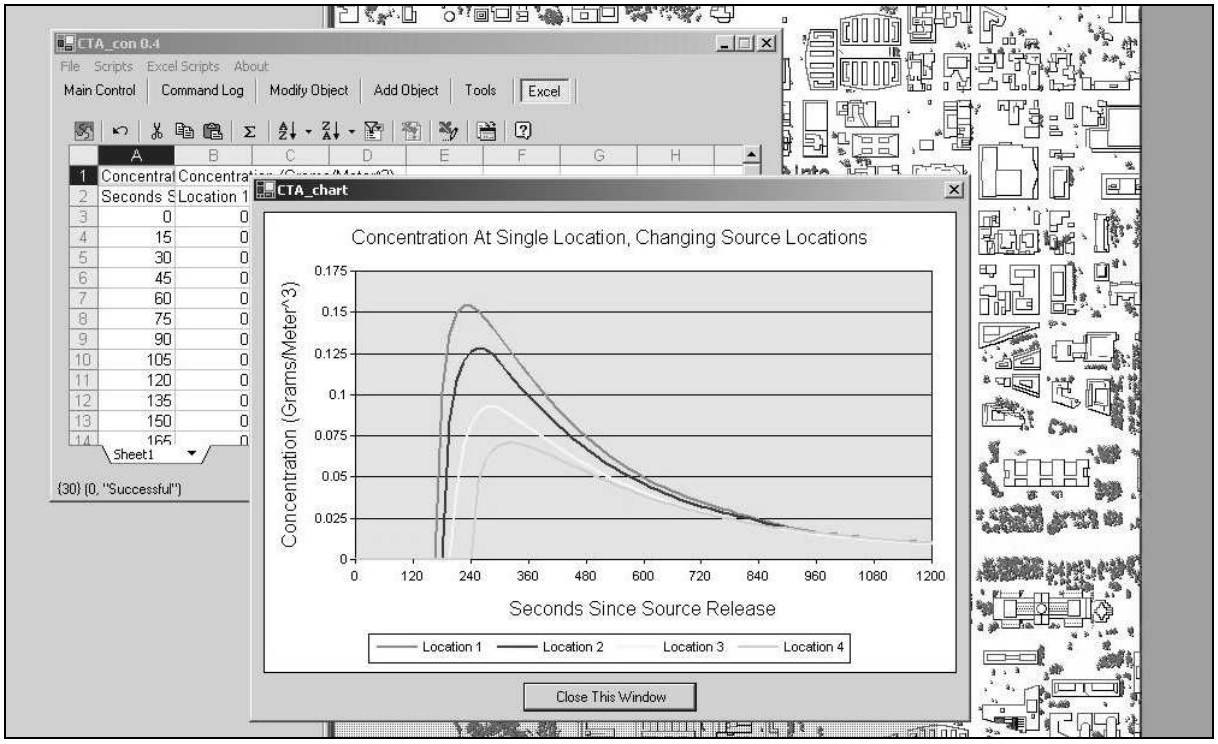


Figure 3: Data Integration With Office

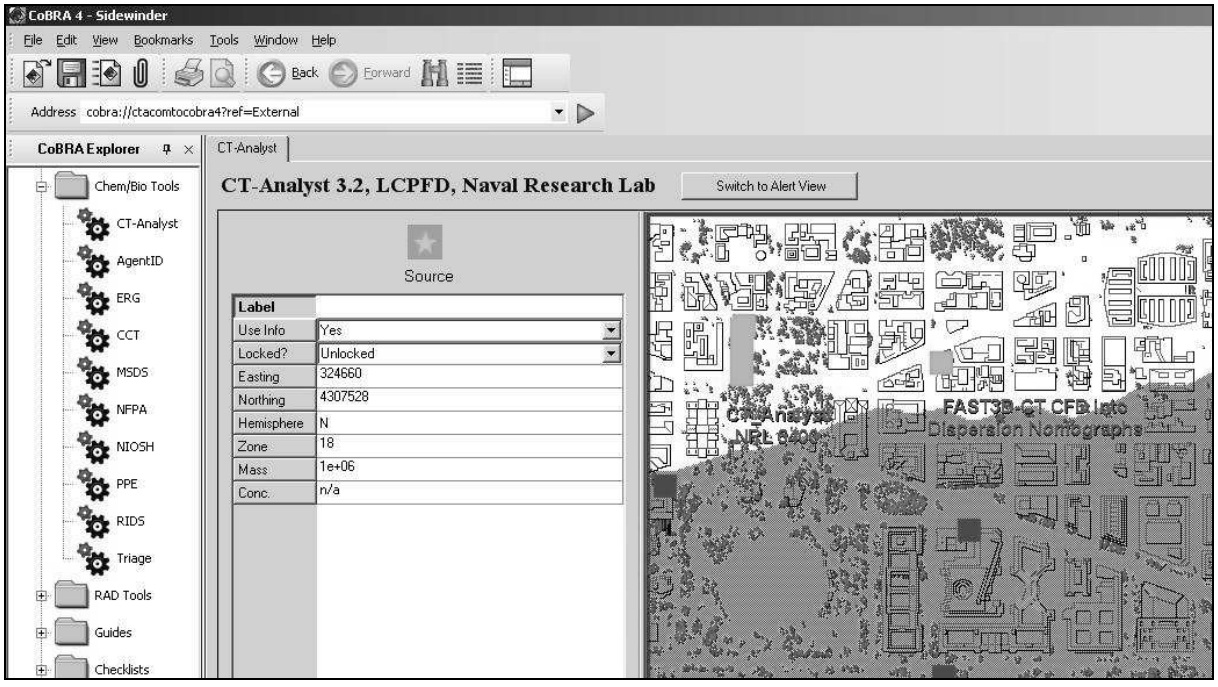
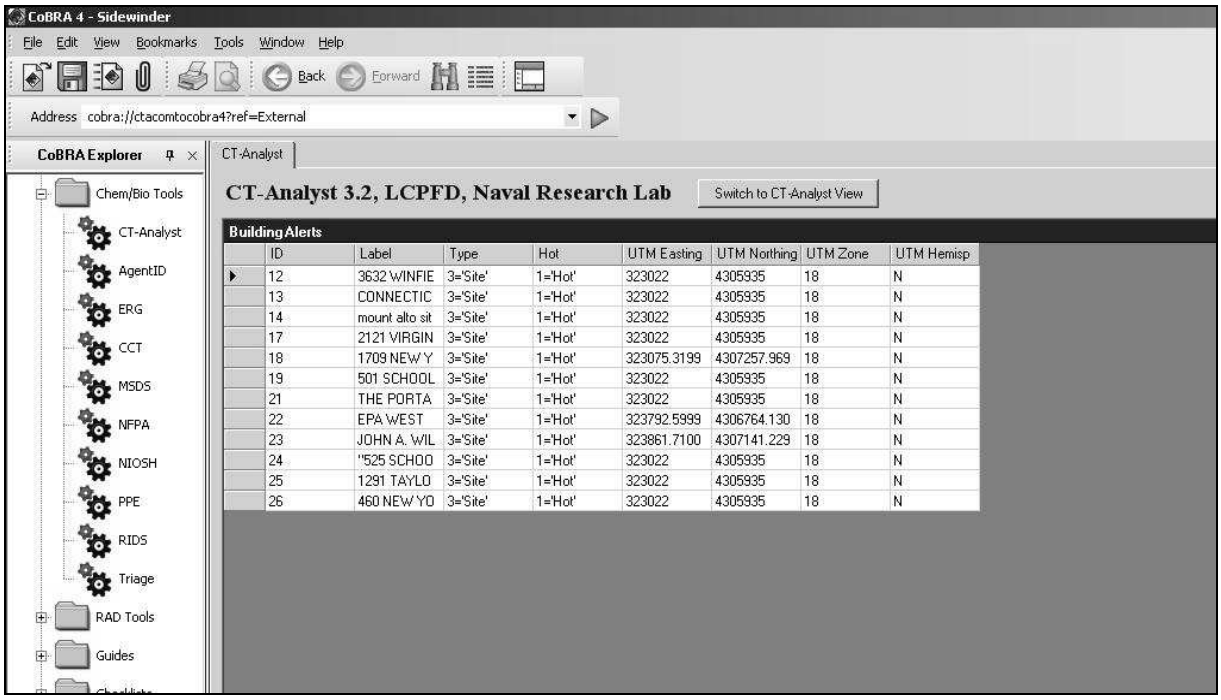


Figure 4: ctaCOM In CoBRA



	ID	Label	Type	Hot	UTM Easting	UTM Northing	UTM Zone	UTM Hemisp
▶	12	3632 WINFIE	3='Site'	1='Hot'	323022	4305935	18	N
	13	CONNECTIC	3='Site'	1='Hot'	323022	4305935	18	N
	14	mount alto sit	3='Site'	1='Hot'	323022	4305935	18	N
	17	2121 VIRGIN	3='Site'	1='Hot'	323022	4305935	18	N
	18	1709 NEW Y	3='Site'	1='Hot'	323075.3199	4307257.969	18	N
	19	501 SCHOOL	3='Site'	1='Hot'	323022	4305935	18	N
	21	THE PORTA	3='Site'	1='Hot'	323022	4305935	18	N
	22	EPA WEST	3='Site'	1='Hot'	323792.5999	4306764.130	18	N
	23	JOHN A. WIL	3='Site'	1='Hot'	323861.7100	4307141.229	18	N
	24	"525 SCHOO	3='Site'	1='Hot'	323022	4305935	18	N
	25	1291 TAYLO	3='Site'	1='Hot'	323022	4305935	18	N
	26	460 NEW YO	3='Site'	1='Hot'	323022	4305935	18	N

Figure 5: ctaCOM Produced Alert View

Figure 6: CT-Analyst footprint outline in ArcGIS on the right, Plotted directly with integrated ctaCOM on left